HMM Specialization with Selective Lexicalization[*]

Jin-Dong Kim and **Sang-Zoo Lee** and **Hae-Chang Rim**
Dept. of Computer Science and Engineering, Korea University,
Anam-dong, Seongbuk-ku, Seoul 136-701, Korea
E-mail: {jin|zoo|rim}@nlp.korea.ac.kr

Abstract

We present a technique which complements Hidden Markov Models by incorporating some lexicalized states representing syntactically uncommon words. Our approach examines the distribution of transitions, selects the uncommon words, and makes lexicalized states for the words. We performed a part-of-speech tagging experiment on the Brown corpus to evaluate the resultant language model and discovered that this technique improved the tagging accuracy by 0.21% at the 95% level of confidence.

1 Introduction

Hidden Markov Models are widely used for statistical language modelling in various fields, e.g., part-of-speech tagging or speech recognition (Rabiner and Juang, 1986). The models are based on Markov assumptions, which make it possible to view the language prediction as a Markov process. In general, we make the first-order Markov assumptions that the current tag is only dependant on the previous tag and that the current word is only dependant on the current tag. These are very ‘strong’ assumptions, so that the first-order Hidden Markov Models have the advantage of drastically reducing the number of its parameters. On the other hand, the assumptions restrict the model from utilizing enough constraints provided by the local context and the resultant model consults only a single category as the contex.

A lot of effort has been devoted in the past to make up for the insufficient contextual information of the first-order probabilistic model. The second order Hidden Markov Models with appropriate smoothing techniques show better performance than the first order models and is considered a state-of-the-art technique (Merialdo, 1994; Brants, 1996). The complexity of the model is however relatively very high considering the small improvement of the performance.

Garside describes IDIOMTAG (Garside et al., 1987) which is a component of a part-of-speech tagging system named CLAWS. IDIOMTAG serves as a front-end to the tagger and modifies some initially assigned tags in order to reduce the amount of ambiguity to be dealt with by the tagger. IDIOMTAG can look at any combination of words and tags, with or without intervening words. By using the IDIOMTAG, CLAWS system improved tagging accuracy from 94% to 96-97%. However, the manual-intensive process of producing idiom tags is very expensive although IDIOMTAG proved fruitful.

Kupiec (Kupiec, 1992) describes a technique of augmenting the Hidden Markov Models for part-of-speech tagging by the use of networks. Besides the original states representing each part-of-speech, the network contains additional states to reduce the noun/adjective confusion, and to extend the context for predicting past participles from preceding auxiliary verbs when they are separated by adverbs. By using these additional states, the tagging system improved the accuracy from 95.7% to 96.0%. However, the additional context is chosen by analyzing the tagging errors manually.

An automatic refining technique for Hidden Markov Models has been proposed by Brants (Brants, 1996). It starts with some initial first order Markov Model. Some states of the model are selected to be split or merged to take into account their predecessors. As a result, each of

[*] The research underlying this paper was supported by research grants from Korea Science and Engineering Foundation.

new states represents a extended context. With this technique, Brants reported a performance equivalent to the second order Hidden Markov Models.

In this paper, we present an automatic refining technique for statistical language models. First, we examine the distribution of transitions of lexicalized categories. Next, we break out the uncommon ones from their categories and make new states for them. All processes are automated and the user has only to determine the extent of the breaking-out.

2 "Standard" Part-of-Speech Tagging Model based on HMM

From the statistical point of view, the tagging problem can be defined as the problem of finding the proper sequence of categories $c_{1,n} = c_1, c_2, ..., c_n$ ($n \geq 1$) given the sequence of words $w_{1,n} = w_1, w_2, ..., w_n$ (We denote the i'th word by w_i, and the category assigned to the w_i by c_i), which is formally defined by the following equation:

$$\mathcal{T}(w_{1,n}) = arg \max_{c_{1,n}} P(c_{1,n}|w_{1,n}) \quad (1)$$

Charniak (Charniak et al., 1993) describes the "standard" HMM-based tagging model as Equation 2, which is the simplified version of Equation 1.

$$\mathcal{T}(w_{1,n}) = arg \max_{c_{1,n}} \prod_{i=1}^{n} P(c_i|c_{i-1})P(w_i|c_i) \quad (2)$$

With this model, we select the proper category for each word by making use of the contextual probabilities, $P(c_i|c_{i-1})$, and the lexical probabilities, $P(w_i|c_i)$. This model has the advantages of a provided theoretical framework, automatic learning facility and relatively high performance. It is thereby at the basis of most tagging programs created over the last few years.

For this model, the first-order Markov assumptions are made as follows:

$$P(c_i|c_{1,i-1}, w_{1,i-1}) \approx P(c_i|c_{i-1}) \quad (3)$$

$$P(w_i|c_{1,i}, w_{1,i-1}) \approx P(w_i|c_i) \quad (4)$$

With Equation 3, we assume that the current category is independent of the previous words and only dependent on the previous category. With Equation 4, we also assume that the correct word is independent of everything except the knowledge of its category. Through these assumptions, the Hidden Markov Models have the advantage of drastically reducing the number of parameters, thereby alleviating the sparse data problem. However, as mentioned above, this model consults only a single category as context and does not utilize enough constraints provided by the local context.

3 Some Refining Techniques for HMM

The first-order Hidden Markov Models described in the previous section provides only a single category as context. Sometimes, this first-order context is sufficient to predict the following parts-of-speech, but at other times (probably much more often) it is insufficient. The goal of the work reported here is to develop a method that can automatically refine the Hidden Markov Models to produce a more accurate language model. We start with the careful observation on the assumptions which are made for the "standard" Hidden Markov Models. With the Equation 3, we assume that the current category is only dependent on the preceding category. As we know, it is not always true and this first-order Markov assumption restricts the disambiguation information within the first-order context.

The immediate ways of enriching the context are as follows:

- to lexicalize the context.
- to extend the context to higher-order.

To lexicalize the context, we include the preceding word into the context. Contextual probabilities are then defined by $P(c_i|c_{i-1}, w_{i-1})$. Figure 1 illustrates the change of dependency when each method is applied respectively. Figure 1(a) represents that each first-order contextual probability and lexical probability are independent of each other in the "standard" Hidden Markov Models, where Figure 1(b) represents that the lexical probability of the preceding word and the contextual probability of the current category are tied into a lexicalized contextual probability.

To extend the context to higher-order, we extend the contextual probability to the second-

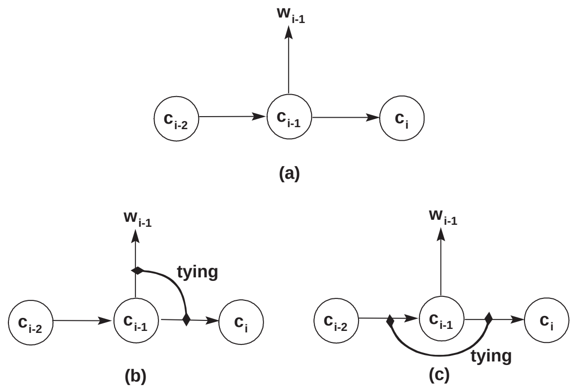


Figure 1: Two Types of Weakening the Markov Assumption

order. Contextual probabilities are then defined by $P(c_i|c_{i-1}, c_{i-2})$. Figure 1(c) represents that the two adjacent contextual probabilities are tied into the second-order contextual probability.

The simple way of enriching the context is to extend or lexicalize it uniformly. The uniform extension of context to the second order is feasible with an appropriate smoothing technique and is considered a state-of-the-art technique, though its complexity is very high: In the case of the Brown corpus, we need trigrams up to the number of 0.6 million. An alternative to the uniform extension of context is the selective extension of context. Brants(Brants, 1996) takes this approach and reports a performance equivalent to the uniform extension with relatively much low complexity of the model.

The uniform lexicalization of context is computationally prohibitively expensive: In the case of the Brown corpus, we need lexicalized bigrams up to the number of almost 3 billion. Moreover, many of these bigrams neither contribute to the performance of the model, nor occur frequently enough to be estimated properly. An alternative to the uniform lexicalization is the selective lexicalization of context, which is the main topic of this paper.

4 Selective Lexicalization of HMM

This section describes a new technique for refining the Hidden Markov Model, which we call selective lexicalization. Our approach automatically finds out syntactically uncommon words and makes a new state (we call it a **lexicalized state**)for each of the words.

Given a fixed set of categories, $\{c^1, c^2, ..., c^C\}$, e.g., $\{adjective, ..., verb\}$, we assume the discrete random variable $\mathbf{X}_{c^j}$ with domain the set of categories and range a set of conditional probabilities. The random variable $\mathbf{X}_{c^j}$ then represents a process of assigning a conditional probability $P(c^i|c^j)$ to every category c^i (c^i ranges over $c^1 ... c^C$)

$$\begin{aligned}\mathbf{X}_{c^j}(c^1) &= P(c^1|c^j)\\ \mathbf{X}_{c^j}(c^2) &= P(c^2|c^j)\\ &...\\ \mathbf{X}_{c^j}(c^C) &= P(c^C|c^j)\end{aligned}$$

We convert the process of $\mathbf{X}_{c^j}$ into the **state transition vector**, $\mathbf{V}_{c^j}$, which consists of the corresponding conditional probabilities, e.g.,

$$\mathbf{V}_{prep} = (P(adjective|prep), ..., P(verb|prep))^T.$$

The (squared) distance between two arbitrary vectors is then computed as follows:

$$R(\mathbf{V}_1, \mathbf{V}_2) = (\mathbf{V}_1 - \mathbf{V}_2)^T(\mathbf{V}_1 - \mathbf{V}_2) \qquad (5)$$

Similarly, we define the **lexicalized state transition vector**[1], $\mathbf{V}_{c^j,w^k}$, e.g.,

$$\mathbf{V}_{prep,in} = (P(adjective|prep, in), ..., P(verb|prep, in))^T.$$

In this situation, it is possible to regard each lexicalized state transition vector, $\mathbf{V}_{c^j,w^k}$, of the same category c^j as members of a cluster whose centroid is the state transition vector, $\mathbf{V}_{c^j}$. We can then compute the deviation of each lexicalized state transition vector, $\mathbf{V}_{c^j,w^k}$, from its corresponding centroid.

$$D(\mathbf{V}_{c^j,w^k}) = (\mathbf{V}_{c^j,w^k} - \mathbf{V}_{c^j})^T(\mathbf{V}_{c^j,w^j} - \mathbf{V}_{c^j}) \quad (6)$$

Figure 2 represents the distribution of lexicalized state transition vectors according to their deviations. As you can see in the figure, the majority of the vectors are near their centroids and only a small number of vectors are very far from their centroids. In the first-order context

[1] To alleviate the sparse data problem, we smoothed the lexicalized state transition probabilities by MacKay and Peto(MacKay and Peto, 1995)'s smoothing technique.

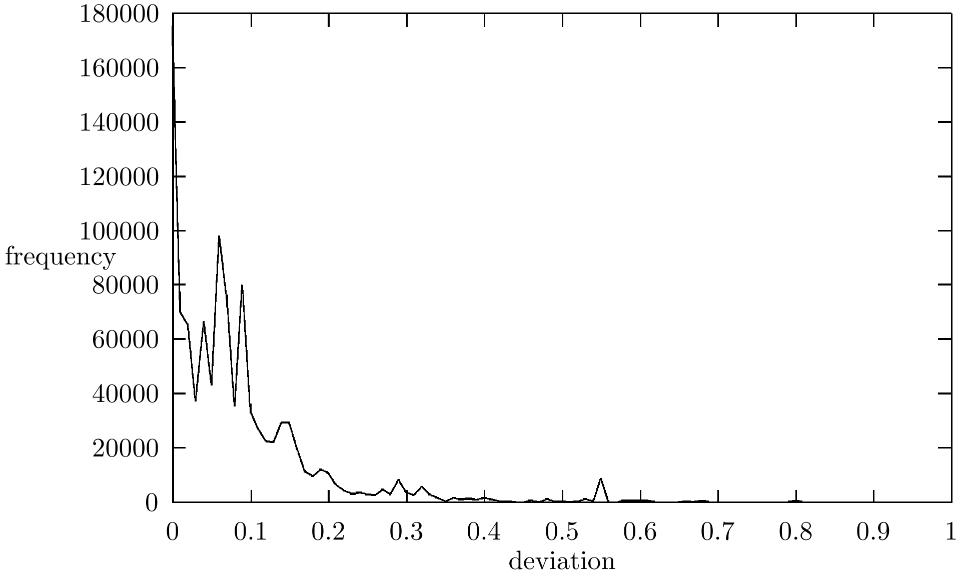


Figure 2: Distribution of Lexicalized Vectors according to Deviation

model (without considering lexicalized context), the centroids represent all the members belonging to it. In fact, the deviation of a vector is a kind of (squared) error for the vector. The error for a cluster is

$$e(\mathbf{V}_{c^j}) = \sum_{w^k} D(\mathbf{V}_{c^j,w^k}) \tag{7}$$

and the error for the overall model is simply the sum of the individual cluster errors:

$$E = \sum_{c^j} e(\mathbf{V}_{c^j}) \tag{8}$$

Now, we could break out a few lexicalized state vectors which have large deviation ($D > \theta$) and make them individual clusters to reduce the error of the given model.

As an example, let's consider the **preposition** cluster. The value of each component of the centroid, $\mathbf{V}_{prep}$, is illustrated in Figure 3(a) and that of the lexicalized vectors, $\mathbf{V}_{prep,in}$, $\mathbf{V}_{prep,with}$ and $\mathbf{V}_{prep,out}$ are in Figure 3(b), (c) and (d) respectively. As you can see in these figures, most of the prepositions including *in* and *with* are immediately followed by article(AT), noun(NN) or pronoun(NP), but the word *out* as preposition shows a completely different distribution. Therefore, it would be a good choice to break out the lexicalized vector, $\mathbf{V}_{prep,out}$, from its centroid, $\mathbf{V}_{prep}$.

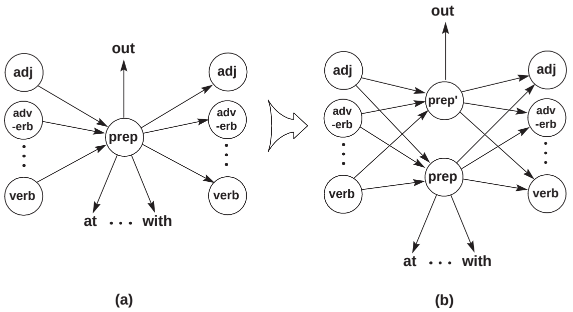
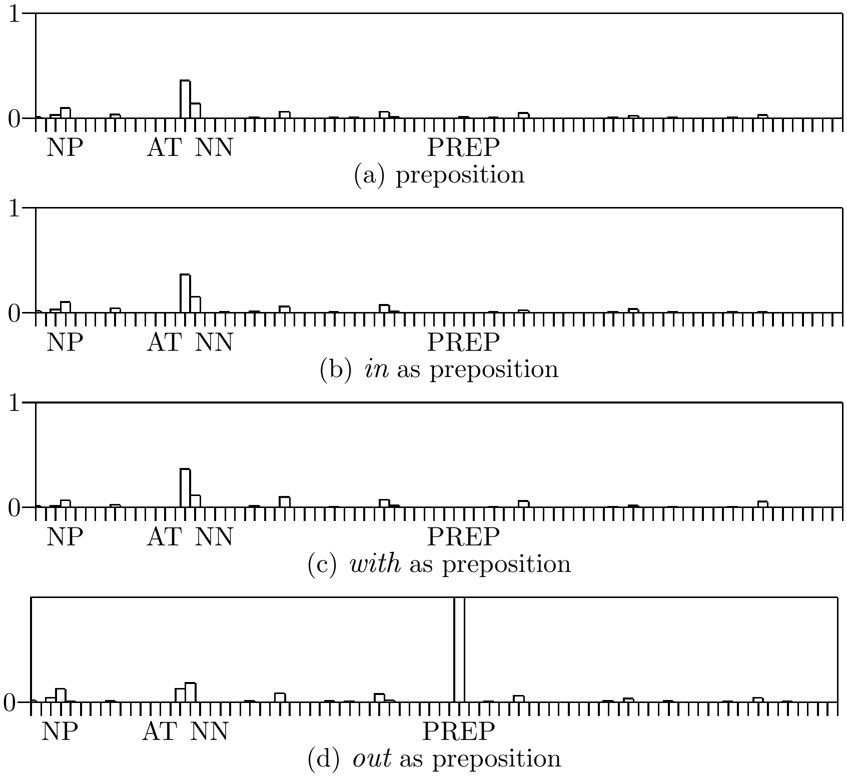


Figure 4: Splitting the **preposition** State

From the viewpoint of a network, the state representing **preposition** is split into two states; the one is the state representing ordinary prepositions except *out*, and the other is the state representing the special preposition *out*, which we call a **lexicalized state**. This process of splitting is illustrated in Figure 4.

Splitting a state results in some changes of the parameters. The changes of the parameters resulting from lexicalizing a word, w^k, in a category, c^j, are indicated in Table 1 (c^i ranges over $c^1 ... c^C$). This full splitting will increase the complexity of the model rapidly, so that estimating the parameters may suffer from the sparseness of the data.

To alleviate it, we use the pseudo splitting



(a) preposition

(b) *in* as preposition

(c) *with* as preposition

(d) *out* as preposition

Figure 3: Transition Vectors in **preposition** Cluster

Table 1: Changes of Parameters in Full Splitting

before splitting	after splitting
$P(w^i \mid c^j)$	$P(w^i \mid c^j, w^k)$ $P(w^i \mid c^j, \neg w^k)$
$P(c^i \mid c^j)$	$P(c^i \mid c^j, w^k)$ $P(c^i \mid c^j, \neg w^k)$
$P(c^j \mid c^i)$	$P(c^j, w^k \mid c^i)$ $P(c^j, \neg w^k \mid c^i)$

Table 2: Changes of Parameters in Pseudo Splitting

before splitting	after splitting
$P(w^i \mid c^j)$	$P(w^i \mid c^j)$
$P(c^i \mid c^j)$	$P(c^i \mid c^j, w^k)$ $P(c^i \mid c^j, \neg w^k)$
$P(c^j \mid c^i)$	$P(c^j \mid c^i)$

which leads to relatively small increment of the parameters. The changes of the parameters in pseudo splitting are indicated in Table 2.

5 Experimental Result

We have tested our technique through part-of-speech tagging experiments with the Hidden Markov Models which are variously lexicalized. In order to conduct the tagging experiments, we divided the whole Brown (tagged) corpus containing 53,887 sentences (1,113,191 words) into two parts. For the **training set**, 90% of the sentences were chosen at random, from which we collected all of the statistical data. We reserved the other 10% for testing. Table 3 lists the basic statistics of our corpus.

Table 3: Overview of Our Corpora

	# of sentences	# of words
training set	48,499	1,001,712
test set	5,388	111,479

We used a tag set containing 85 categories. The amount of ambiguity of the test set is summarized in Table 4. The second column shows that words to the ratio of 52% (the number of 57,808) are not ambiguous. The tagger attempts to resolve the ambiguity of the remaining words.

Table 4: Amount of Ambiguity of Test Set

ambiguity(#)	1	2	3	4	5	6	total
ratio(%)	52	30	8	7	2	1	100

Figure 5 and Figure 6 show the results of our part-of-speech tagging experiments with the "standard" Hidden Markov Model and variously lexicalized Hidden Markov Models using full splitting method and pseudo splitting method respectively.

We got 95.7858% of the tags correct when we applied the standard Hidden Markov Model without any lexicalized states. As the number of lexicalized states increases, the tagging accuracy increases until the number of lexicalized states becomes 160 (using full splitting) and 210 (using pseudo splitting). As you can see in these figures, the full splitting improves the performance of the model more rapidly but suffer more sevelery from the sparseness of the training data. In this experiment, we employed Mackay and Peto's smoothing techniques for estimating the parameters required for the models. The best precision has been found to be 95.9966% through the model with the 210 lexcalized states using the pseudo splitting method.

6 Conclusion

In this paper, we present a method for complementing the Hidden Markov Models. With this method, we lexicalize the Hidden Markov Model seletively and automatically by examining the transition distribution of each state relating to certain words.

Experimental results showed that the selective lexicalization improved the tagging accurary from about 95.79% to about 96.00%. Using normal tests for statistical significance we found that the improvement is significant at the 95% level of confidence.

The cost for this improvement is minimal. The resulting network contains 210 additional lexicalized states which are found automatically. Moreover, the lexicalization will not decrease the tagging speed[2], because the lexicalized states and their corresponding original states are exclusive in our lexicalized network, and thus the rate of ambiguity is not increased even if the lexicalized states are included.

Our approach leaves much room for improvement. We have so far considered only the outgoing transitions from the target states. As a result, we have discriminated only the words with right-associativity. We could also discriminate the words with left-associativity by examining the incoming transitions to the state. Furthermore, we could extend the context by using the second-order context as represented in Figure 1(c). We believe that the same technique presented in this paper could be applied to the proposed extensions.

References

T. Brants. 1996. Estimating markov model structures. In *Proceedings of the Fourth International Conference on Spoken Language Processing*, pages 893–896.

E. Charniak, C. Hendrickson, N. Jacobson, and M. Perkowitz. 1993. Equations for part-of-speech tagging. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 784–789.

K. Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 136–143.

S. Derose. 1988. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):31–39.

R. Garside, G. Leech, and G. Sampson. 1987. *The Computational Analysis of English.* Longman Group.

J. Kupiec. 1992. Robust part-of-speech tagging using a hidden markov model. *Computer Speech and Language*, 6:225–242.

D. MacKay and L. Peto. 1995. A hierarchical dirichlet language model. *Natural Language Engineering*, 1(3):289–307.

[2] The Viterbi algorithm for finding the best tags runs in $O(n^2)$ where n is the number of states.

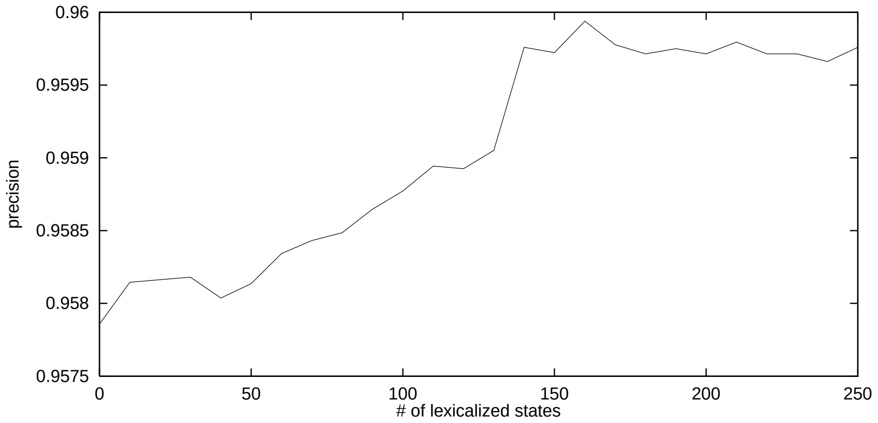


Figure 5: POS tagging results with lexicalized HMM using full splitting method

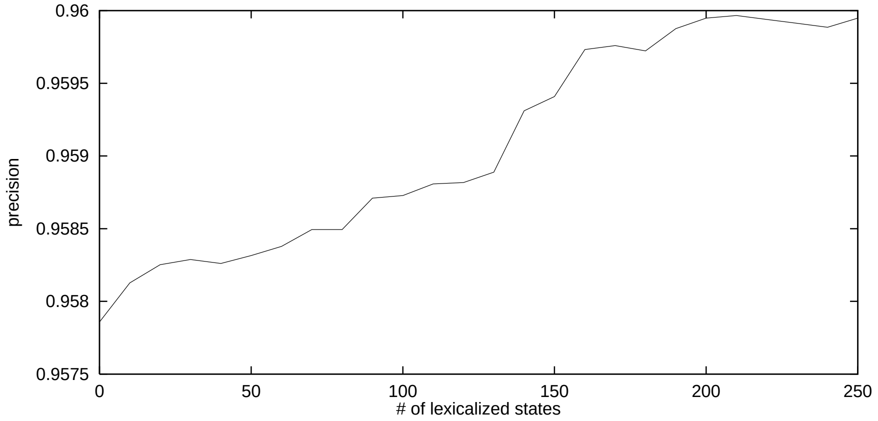


Figure 6: POS tagging results with lexicalized HMM using pseudo splitting method

B. Merialdo. 1994. Tagging english text with a probabilistic model. *Computational Linguistics*, 20(4):155–171.

L. Rabiner and B. Juang. 1986. An introduction to hidden markov models. *IEEE ASSP Magazine*, pages 4–16, January.

Appendix :

Top 100 words with high deviation

according(IN) rather(IN) out(IN) able(JJ) Aj(NP) no(QL) because(RB) however(RB) as(IN) per(IN) trying(VBG) however(WRB) likely(JJ) United(NP) Mrs.(NP) New(NP) Rhode(NP) National(NP) Miss(NP) tried(VBD) Dr.(NP) lack(NN) much(QL) Mr.(NP) North(NP) June(NP) A.(NP) J.(NP) right(QL) May(NP) ready(JJ) St.(NP) even(QL) various(JJ) don't(DO) instead(RB) far(QL) B(NP) didn't(DOD) try(VB) available(JJ) William(NP) !(.) ?(.) ;(.) number(NN) so(CS) due(JJ) World(NP) Christian(NP) difficult(JJ) tell(VB) going(VBG) kind(NN) let(VB) continue(VB) series(NN) part(NN) radio(NN) sure(JJ) want(VB) front(NN) seem(VB) total(NN) decided(VBD) expected(VBN) right(NN) based(VBN) White(NP) except(IN) told(VBD) James(NP) fact(NN) March(NP) sort(NN) example(NN) designed(VBN) respect(NN) talk(VB) Department(NP) single(AP) Negro(NP) wanted(VBD) Western(NP) yes(RB) become(VBN) necessary(JJ) speak(VB) about(RB) amount(NN) down(IN) like(VB) S.(NP) same(AP) too(RB) General(NP) began(VBD) use(NN) tax(NN) got(VBN)